Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	614	(namespace or (name adj space)) same (file adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 12:10
L2	4	1 and redirect$6 near2 message	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 12:12
L3	144	1 and redirect$6 and location	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 12:12
L4	36	1 and (redirect$6 and location) same (file adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 12:12

Google Web Images Groups News Froogle Local[New!] more »
"file system" +redirection +encode +message Search Advanced Search Preferences

Web Results **1** - **10** of about **12,400** for **"file system" +redirection +encode +message**. (**0.23** seconds)

Security Glossary: Audits ...
... and to provide the receiver with the means to **encode** a reply. ... CIFS/SMB or Common Internet **File System**/Server **Message** Block is the under pinning file ...
www.iioo.co.uk/Gloss/security.htm - 31k - Cached - Similar pages

Batch Scripts for Windows
... self-contained procedures **message**/dialog boxes looping (iterative) logic ... command-line tool (screnc.exe) download to **encode** (not really compile) ...
www.wilsonmar.com/1wsh.htm - 49k - Cached - Similar pages

Commands AM
... Public Domain utility that sends the contents of a file in an e-mail **message** using SMTP ... FSUTIL Perform many FAT and NTFS **file system** related tasks, ...
home.earthlink.net/~rlively/MANUALS/INDEXA.HTM - 52k - Cached - Similar pages

Products
... SMAPIweb makes heavy use of the POSIX **file system** to provide an effective ... with **redirection** directly from a job and send this data in a mail **message**, ...
www.emailinc.com/hpproducts.htm - 101k - Cached - Similar pages

Quick Unix Reference
... file **encode** file (compression #) ==> file.gz compress file **encode** file, ... 3.4 **Redirection**. command > file direct output of command to file instead of ...
cdfrh0.grid.umich.edu/~claudiof/Tools/unix/quick.html - 12k - Apr 18, 2005 - Cached - Similar pages

Unix Quick Reference
... gzip file **encode** file, replacing it with file.gz gzip -d file.gz decode ... show show current **message** next show next **message** prev show previous **message** ...
www.cs.utk.edu/~help/unix/quickref.php - 16k - Cached - Similar pages

Software Download: Http
... Find out what sort of URL **redirection** the site may be using. ... parse HTML tags and **encode** / decode any content using Base64, UUE and Quoted Printable ...
www.sharewareconnection.com/titles/http.htm - 91k - Cached - Similar pages

[DOC] Bilkent University
File Format: Microsoft Word 2000 - View as HTML
... SSL (Secure Sockets Layer): A 128 bit encryption system used to **encode** the ... The user is redirected with client-side **redirection** to the originally ...
www.ug.bcc.bilkent.edu.tr/ ~erem/High%20Level%20Design.doc - Apr 18, 2005 - Similar pages

SunOS man pages
... space used for caching file systems with the Cache **File-System** (CacheFS) ... STREAMS data structure for the M_COPYIN and the M_COPYOUT **message** types ...
docsun.cites.uiuc.edu/sun_docs/ C/solaris_9/SUNWaman/c.html - 70k - Cached - Similar pages

Glossary - Windows Server 2003
... software installation, scripts, folder **redirection**, remote installation ...

In Distributed **File System** (DFS), replication synchronizes files and folders ...
www.microsoft.com/Resources/Documentation/ windowsserv/2003/all/techref/en-us/gloss_rktools.asp - 164k - Cached - Similar pages

Gooooooooogle ▶

Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

"file system" +redirection +encode + Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Google Web Images Groups News Froogle Local[New!] more »
"file system" +redirection +encode +message Search Advanced Search Preferences

Web Results **11** - **20** of about **12,400** for **"file system" +redirection +encode +message**. (**0.06** seconds)

Sponsored Links

File System
Article in PC Magazine
Read it online. Free Trial!
www.KeepMedia.com

Software Download: Http 404
... Find out what sort of URL **redirection** the site may be using. ... decodes compressed entities, and saves entities in your **file system**. ...
www.sharewareconnection.com/titles/http-404.htm - 84k - Cached - Similar pages

Programming for Internationalization FAQ
... the **message** catalog "helloprogram" printf(dgettext("helloprogram","Hello, ... Thus, even simple operations such as 'mv', 'cp' or file **redirection** may be ...
www.cs.uu.nl/wais/html/na-dir/ internationalization/programming-faq.html - 34k - Cached - Similar pages

[PDF] Synchronous IPC over Transparent Monitors
File Format: PDF/Adobe Acrobat - View as HTML
... tomized IPC, but the **redirection** is not transparent to the ... delivered to the new secret **file system**. The notification. **message** contains the identity ...
i30www.ira.uka.de/research/ documents/l4ka/2000/synchronous-ipc.pdf - Apr 18, 2005 - Similar pages

SIPfoundry SIP softphone architecture
... Dejitter, Decode, **Encode**, and ToNet resources are added to the flow graph. ... The proxy is concerned with the routing and **redirection** of the **message**. ...
www.sipfoundry.org/sipXphone/softphone_arch.html - 50k - Cached - Similar pages

Code Style: Java servlets frequently asked questions (FAQ)
... What's the difference between forward , include and **redirection**? ... Provided all subsequent pages **encode** their links too, a URL session can be passed ...
www.codestyle.org/java/servlets/FAQ.shtml - 50k - Cached - Similar pages

Unix KornShell Quick Reference
... crontab file crypt **encode**/decode files csh a shell (command interpreter) ... NFS file systems backup backup or archive **file system** bdf report number of ...
www.maththinking.com/boat/kornShell.html - 76k - Cached - Similar pages

RFC 3986
... URI producing applications should percent-**encode** data octets that ... use of **redirection** by HTTP origin servers to avoid problems with relative ...
asg.web.cmu.edu/rfc/rfc3986.html - 149k - Cached - Similar pages

Exim 4.50 Specification chapter 26
... name) of the file or directory generated by the **redirection** operation. ... on the operating system (and possibly the **file system**) that is being used. ...
www.exim.org/exim-html-4.50/doc/html/spec_26.html - 60k - Cached - Similar pages

manquery - @ Eastern Illinois University
... canputnext (9f), canputnext - test for room in next module's **message** queue ... space used for caching file systems with the Cache **File-System** (CacheFS) ...
www.eiu.edu/cgi-bin/manquery?+C - 93k - Cached - Similar pages

I Want My FTP: Bits on Demand
... For example, it has no way to **encode** a "library" (ie, a named collection of ... to be more

about the **file system** than about the data connection?" ...
www.4k-associates.com/4K-Associates/IEEE-L7-FTP.html - 29k - Cached - Similar pages

◀ Goooooooooogle ▶
Result Page: **Previous** 1 **2** 3 4 5 6 7 8 9 10 11 **Next**

"file system" +redirection +encode + Search

Search within results | Language Tools | Search Tips

Google Home - Advertising Programs - Business Solutions - About Google

Terms used **file system redirection message encode** Found **1,346** of **153,034**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Fast and scalable wireless handoffs in supports of mobile Internet audio

Ramón Cáceres, Venkata N. Padmanabhan

December 1998 **Mobile Networks and Applications**, Volume 3 Issue 4

Full text available: pdf(187.08 KB) Additional Information: full citation, abstract, references, citings, index terms

Future internetworks will include large numbers of portable devices moving among small wireless cells. We propose a hierarchical mobility management scheme for such networks. Our scheme exploits locality in user mobility to restrict handoff processing to the vicinity of a mobile node. It thus reduces handoff latency and the load on the internetwork. Our design is based on the Internet Protocol (IP) and is compatible with the Mobile IP standard. We also present experimental results for the I ...

2 Alphabet Soup

Stephen Turnbull

March 1999 **Linux Journal**

Full text available: html(40.76 KB) Additional Information: full citation, abstract, references, index terms

The Internationalization of Linux, Part 1: Mr. Turnbull takes a look at the problems faced when different character sets and the need for standardization

3 Compactly encoding unstructured inputs with differential compression

Miklos Ajtai, Randal Burns, Ronald Fagin, Darrell D. E. Long, Larry Stockmeyer

May 2002 **Journal of the ACM (JACM)**, Volume 49 Issue 3

Full text available: pdf(348.32 KB) Additional Information: full citation, abstract, references, citings, index terms

The subject of this article is *differential compression*, the algorithmic task of finding common strings between versions of data and using them to encode one version compactly by describing it as a set of changes from its companion. A main goal of this work is to present new differencing algorithms that (i) operate at a fine granularity (the atomic unit of change), (ii) make no assumptions about the format or alignment of input data, and (iii) in practice use linear time, use constant spa ...

Keywords: Delta compression, differencing, differential compression

4 Industrial sessions: beyond relational tables: Coordinating backup/recovery and data

consistency between database and file systems

Suparna Bhattacharya, C. Mohan, Karen W. Brannon, Inderpal Narang, Hui-I Hsiao, Mahadevan Subramanian

June 2002 **Proceedings of the 2002 ACM SIGMOD international conference on Management of data**

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, index terms

Managing a combined store consisting of database data and file data in a robust and consistent manner is a challenge for database systems and content management systems. In such a hybrid system, images, videos, engineering drawings, etc. are stored as files on a file server while meta-data referencing/indexing such files is created and stored in a relational database to take advantage of efficient search. In this paper we describe solutions for two potentially problematic aspects of such a data ...

Keywords: DB2, content management, database backup, database recovery, datalinks

5 Mobile data management: Mimic: raw activity shipping for file synchronization in mobile file systems

Tae-Young Chang, Aravind Velayutham, Raghupathy Sivakumar

June 2004 **Proceedings of the 2nd international conference on Mobile systems, applications, and services**

Full text available: pdf(334.54 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we consider the problem of file synchronization when a mobile host shares files with a backbone file server in a network file system. Several *diff* schemes have been proposed to improve upon the transfer overheads of conventional file synchronization approaches which use full file transfer. These schemes compute the binary *diff* of the new file with respect to the old copy at the server and transfer the computed *diff* to the server for file-synchronization. Howev ...

Keywords: file synchronization, mobile file system, raw activity shipping

6 ASN.1 protocol specification for use with arbitrary encoding schemes

Duke Tantiprasut, John Neil, Craig Farrell

August 1997 **IEEE/ACM Transactions on Networking (TON)**, Volume 5 Issue 4

Full text available: pdf(159.77 KB) Additional Information: full citation, references, citings, index terms

7 File and storage systems: Decentralized user authentication in a global file system

Michael Kaminsky, George Savvides, David Mazieres, M. Frans Kaashoek

October 2003 **Proceedings of the nineteenth ACM symposium on Operating systems principles**

Full text available: pdf(144.43 KB) Additional Information: full citation, abstract, references, index terms

The challenge for user authentication in a global file system is allowing people to grant access to specific users and groups in remote administrative domains, without assuming any kind of pre-existing administrative relationship. The traditional approach to user authentication across administrative domains is for users to prove their identities through a chain of certificates. Certificates allow for general forms of delegation, but they often require more infrastructure than is necessary to sup ...

Keywords: ACL, SFS, authentication, authorization, credentials, file system, groups, users

8 Access Control Models and Mechanisms: Cryptographic access control in a distributed file system

Anthony Harrington, Christian Jensen

June 2003 **Proceedings of the eighth ACM symposium on Access control models and technologies**

Full text available: pdf(249.24 KB) Additional Information: full citation, abstract, references, index terms

Traditional access control mechanisms rely on a reference monitor to mediate access to protected resources. Reference monitors are inherently centralized and existing attempts to distribute the functionality of the reference monitor suffer from problems of scalability.Cryptographic access control is a new distributed access control paradigm designed for a global federation of information systems. It defines an implicit access control mechanism, which relies exclusively on cryptography to provide ...

Keywords: access control, cryptography, network file systems

9 An efficient bitmap encoding scheme for selection queries

Chee-Yong Chan, Yannis E. Ioannidis

June 1999 **ACM SIGMOD Record , Proceedings of the 1999 ACM SIGMOD international conference on Management of data**, Volume 28 Issue 2

Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, citings, index terms

Bitmap indexes are useful in processing complex queries in decision support systems, and they have been implemented in several commercial database systems. A key design parameter for bitmap indexes is the encoding scheme, which determines the bits that are set to 1 in each bitmap in an index. While the relative performance of the two existing bitmap encoding schemes for simple selection queries of the form "v1 ≤ A

10 Potential benefits of delta encoding and data compression for HTTP

Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, Balachander Krishnamurthy

October 1997 **ACM SIGCOMM Computer Communication Review , Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication**, Volume 27 Issue 4

Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Caching in the World Wide Web currently follows a naive model, which assumes that resources are referenced many times between changes. The model also provides no way to update a cache entry if a resource does change, except by transferring the resource's entire new value. Several previous papers have proposed updating cache entries by transferring only the differences, or "delta," between the cached entry and the current value.In this paper, we make use of dynamic traces of the full contents of ...

11 A low-bandwidth network file system

Athicha Muthitacharoen, Benjie Chen, David Mazières

October 2001 **ACM SIGOPS Operating Systems Review , Proceedings of the eighteenth ACM symposium on Operating systems principles**, Volume 35 Issue 5

Full text available: pdf(1.29 MB) Additional Information: full citation, abstract, references, citings, index terms

Users rarely consider running network file systems over slow or wide-area networks, as the performance would be unacceptable and the bandwidth consumption too high. Nonetheless, efficient remote file access would often be desirable over such networks---particularly when high latency makes remote login sessions unresponsive. Rather than run interactive programs such as editors remotely, users could run the programs locally and manipulate remote files through the file system. To do so, however, wo ...

12 Separating key management from file system security

David Mazières, Michael Kaminsky, M. Frans Kaashoek, Emmett Witchel

December 1999 **ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles**, Volume 33 Issue 5

Full text available: pdf(1.77 MB) Additional Information: full citation, abstract, references, citings, index terms

No secure network file system has ever grown to span the Internet. Existing systems all lack adequate key management for security at a global scale. Given the diversity of the Internet, any particular mechanism a file system employs to manage keys will fail to support many types of use.We propose separating key management from file system security, letting the world share a single global file system no matter how individuals manage keys. We present SFS, a secure file system that avoids internal ...

13 The ITC distributed file system: principles and design

M. Satyanarayanan, John H. Howard, David A. Nichols, Robert N. Sidebotham, Alfred Z. Spector, Michael J. West

December 1985 **ACM SIGOPS Operating Systems Review , Proceedings of the tenth ACM symposium on Operating systems principles**, Volume 19 Issue 5

Full text available: pdf(1.01 MB) Additional Information: full citation, references, citings, index terms

14 Efficient distributed backup with delta compression

Randal C. Burns, Darrell D. E. Long

November 1997 **Proceedings of the fifth workshop on I/O in parallel and distributed systems**

Full text available: pdf(1.37 MB) Additional Information: full citation, references, citings, index terms

15 Architectural considerations for next generation file systems

Prashant Shenoy, Pawan Goyal, Harrick M. Vin

October 1999 **Proceedings of the seventh ACM international conference on Multimedia (Part 1)**

Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

We evaluate two architectural alternatives—partitioned and integrated—for designing next generation file systems. Whereas a partitioned server employs a separate file system for each application class, an integrated file server multiplexes its resources among all application classes; we evaluate the performance of the two architectures with respect to sharing of disk bandwidth among the application classes. We show that although the problem of sharing disk bandwidth in integrate ...

16 TSIOPAK—a proposal for a new Sharp APL file system

Carlos G. Leon

January 1987 **ACM SIGAPL APL Quote Quad , Proceedings of the international conference on APL: APL in transition**, Volume 17 Issue 4

Full text available: pdf(1.63 MB) Additional Information: full citation, abstract, index terms

Communication between OS files is often necessary when information that is manipulated and stored in Sharp APL files is required to be used by non-APL routines. For example, one may use APL for number crunching and have the output processed by SAS or FORTRAN to plot the data. The arising demand for procedures that would make data accessible from APL

to other systems motivated the author to take action. TSIOPAK was developed with the intention to provide a standard inte ...

17 An "open" oriented file system

Bill Mahoney

January 1994 **ACM SIGOPS Operating Systems Review**, Volume 28 Issue 1

Full text available: pdf(395.09 KB) Additional Information: full citation, abstract, citings, index terms

A custom application involving a high percentage of open and close file system calls has provided incentive to design and implement a new file system. The fact that many files are small, and that files are referenced by number instead of by name has allowed several design decisions to be made in an attempt to improve performance. The design of the file system is described, and a comparison is made between the new system and a more traditional system to see if the design has been successful.

18 Software-Only Bus Encoding Techniques for an Embedded System

Wei-Chung Cheng, Jian-Lin Liang, Massoud Pedram

January 2002 **Proceedings of the 2002 conference on Asia South Pacific design automation/VLSI Design**

Full text available: pdf(174.60 KB) Publisher Site Additional Information: full citation, abstract

Microprocessors with built-in Liquid Crystal Device (LCD) controllers and equipped with Flash memory are common in mobile computing applications. In the first part of the paper, a software-only encoding technique is proposed to reduce the power consumption of the processor-memory bus when displaying an image on the LCD. Based on the translation mechanism of the LCD controller, our approach is to start with the palette as a coding table for the pixel buffer and then reassign the codes according t ...

Keywords: memory bus encoding, low power, bus activity minimization, LCD, CompactFlash, Flash memory

19 The design and implementation of tripwire: a file system integrity checker

Gene H. Kim, Eugene H. Spafford

November 1994 **Proceedings of the 2nd ACM Conference on Computer and communications security**

Full text available: pdf(1.22 MB) Additional Information: full citation, abstract, references, citings, index terms

At the heart of most computer systems is a file system. The file system contains user data, executable programs, configuration and authorization information, and (usually) the base executable version of the operating system itself. The ability to monitor file systems for unauthorized or unexpected changes gives system administrators valuable data for protecting and maintaining their systems. However, in environments of many networked heterogeneous platforms with different policies and softw ...

20 Decentralized storage systems: Ivy: a read/write peer-to-peer file system

Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, Benjie Chen

December 2002 **ACM SIGOPS Operating Systems Review**, Volume 36 Issue SI

Full text available: pdf(1.65 MB) Additional Information: full citation, abstract, references

Ivy is a multi-user read/write peer-to-peer file system. Ivy has no centralized or dedicated components, and it provides useful integrity properties without requiring users to fully trust either the underlying peer-to-peer storage system or the other users of the file system.An Ivy file system consists solely of a set of logs, one log per participant. Ivy stores its logs in

the DHash distributed hash table. Each participant finds data by consuiting all logs, but performs modifications by appendi ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	28443	file adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 09:08
L2	48	1 and ((file adj system) near4 server) and (redirect$4 near4 message)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 09:08
L3	22	2 and encod$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/20 09:09

Architectural considerations for next generation file systems

Full text Pdf (1.36 MB)

Source **International Multimedia Conference** archive
Proceedings of the seventh ACM international conference on Multimedia (Part 1) table of contents
Orlando, Florida, United States
Pages: 457 - 467
Year of Publication: 1999
ISBN:1-58113-151-8

Authors
Prashant Shenoy — Department of Computer Science, University of Massachusetts, Amherst, MA
Pawan Goyal — Ensim Corporation, 1215 Terra Bella Ave, Mountain View, CA and Department of Computer Sciences, University of Texas, Austin, TX
Harrick M. Vin — Department of Computer Sciences, University of Texas, Austin, TX

Sponsors SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques
SIGCOMM: ACM Special Interest Group on Data Communication
SIGMULTIMEDIA: ACM Special Interest Group on Multimedia

Publisher ACM Press New York, NY, USA

↑ ABSTRACT

We evaluate two architectural alternatives—partitioned and integrated—for designing next generation file systems. Whereas a partitioned server employs a separate file system for each application class, an integrated file server multiplexes its resources among all application classes; we evaluate the performance of the two architectures with respect to sharing of disk bandwidth among the application classes. We show that although the problem of sharing disk bandwidth in integrated file systems is conceptually similar to that of sharing network link bandwidth in integrated services networks, the arguments that demonstrate the superiority of integrated services networks over separate networks are not applicable to file systems. Furthermore, we show that: (i) an integrated server outperforms the partitioned server in a large operating region and has slightly worse performance in the remaining region, (ii) the capacity of an integrated server is larger than that of the partitioned server, and (iii) an integrated server outperforms the partitioned server by up to a factor of 6 in the presence of bursty workloads.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 P. Baxham. A Fresh Approach to File System Quality of Service. In Proceedings of NOSSDAV'97, $~. Louis, Missouri, pages 119-128, May 1997.

2 Jon C. R. Bennett , Hui Zhang, Hierarchical packet fair queueing algorithms, Conference proceedings on Applications, technologies, architectures, and protocols for computer communications, p.143-156, August 28-30, 1996, Palo Alto, California, United States

3 E G. Coffman and M. Hofri. On the Expected Performance of Scanning Disks. SIAM Journal of Computing, 10(1):60-70, February 1982.

4 E G. Coffman, L A. Klimko, and B. Ryan. Analysis of Scanning Policies for Reducing Disk Seek Times. SIAM Journal of Computing, 1(3):269-279, September 1972.

5 Michael D. Dahlin , Clifford J. Mather , Randolph Y. Wang , Thomas E. Anderson , David A. Patterson, A quantitative analysis of cache policies for scalable network file systems, Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems, p.150-160, May 16-20, 1994, Nashville, Tennessee, United States

6 A. Demers , S. Keshav , S. Shenker, Analysis and simulation of a fair queueing algorithm, Symposium proceedings on Communications architectures & protocols, p.1-12, September 25-27, 1989, Austin, Texas, United States

7 P J. Denning. Effects of Scheduling on File Memory Operations. In Proceedings of AFIP$ SJCC, pages 9- 21, 1967.

8 Robert Geist , Stephen Daniel, A continuum of disk scheduling algorithms, ACM Transactions on Computer Systems (TOCS), v.5 n.1, p.77-92, Feb. 1987

9 Leana Golubchik , John C. S. Lui , Richard Muntz, Reducing I/O demand in video-on-demand storage servers, Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p.25-36, May 15-19, 1995, Ottawa, Ontario, Canada

10 Steven D. Gribble , Gurmeet Singh Manku , Drew Roselli , Eric A. Brewer , Timothy J. Gibson , Ethan L. Miller, Self-similarity in file systems, Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p.141-150, June 22-26, 1998, Madison, Wisconsin, United States

11 Micha Hofri, Disk scheduling: FCFS vs.SSTF revisited, Communications of the ACM, v.23 n.11, p.645-653, Nov. 1980

12 Edward K. Lee , Randy H. Katz, An analytic performance model of disk arrays, Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems, p.98-109, May 10-14, 1993, Santa Clara, California, United States

13 C. Martin, P. S. Narayan, B. Ozden, R. Rastogi, and A. Silberschatz. The FeUini Multimedia Storage Server. Multimedia Information Storage and Management, Editor S. M. Chung, Kluwer Academic Publishers, 1996.

14 Marshall K. McKusick , William N. Joy , Samuel J. Leffler , Robert S. Fabry, A fast file system for UNIX, ACM Transactions on Computer Systems (TOCS), v.2 n.3, p.181-197, Aug. 1984

15 G. Nerjes , P. Muth , M. Paterakis , Y. Romboyannakis , P. Triantafillou , G. Weikum, Scheduling Strategies for Mixed Workloads in Multimedia Information Servers, Proceedings of the Workshop on Research Issues in Database Engineering, p.121, February 23-24, 1998

16 A. L. Narasimha Reddy , Jim Wyllie, Disk scheduling in a multimedia I/O system, Proceedings of the first ACM international conference on Multimedia, p.225-233, August 02-06, 1993, Anaheim, California, United States

17 Timothy Roscoe. The Structure of a Multi-Service Operating System. PhD thesis, University of Cambridge Computer Laboratory, April 1995. Available as Technical Report No. 376.

18 S. Shenker. Fundamental Design Issues for the Future Internet. IEEE Journal of Selected Areas in Communi. cations, 13:1176-1188, September 1995.

19 P. Shenoy , P. Goyal , H. Vin, Architectural Considerations for Next Generation File Systems TITLE2:, University of Massachusetts, Amherst, MA, 1998

20 Prashant J. Shenoy , Harrick M. Vin, Cello: a disk scheduling framework for next generation operating systems, Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p.44-55, June 22-26, 1998, Madison, Wisconsin, United States

21 P J. Shenoy, P. Goyal, S S. Rao, and I-I M. Vin. Symphony: An Integrated Multimedia File System. In Proceedings of the SPIE/A CM Conference on Multimedia Computing and Networking (MMCN'98), San Jose, CA, pages 124-138, January 1998.

22 P j. Shenoy and H M. Vin. Efficient Striping Techniques for Multimedia File Servers. In Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Loins, MO, pages 25-36, May 1997.

23 Toby J. Teorey , Tad B. Pinkerton, A comparative analysis of disk scheduling policies, Communications of the ACM, v.15 n.3, p.177-184, March 1972

24 Fouad A. Tobagi , Joseph Pang , Randall Baird , Mark Gang, Streaming RAID: a disk array management system for video files, Proceedings of the first ACM international conference on Multimedia, p.393-400, August 02-06, 1993, Anaheim, California, United States

25 Athanassios K. Tsiolis , Mary K. Vernon, Group-guaranteed channel capacity in multimedia storage servers, Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, p.285-297, June 15-18, 1997, Seattle, Washington, United States

26 P~. Wijayaratne and A. L. N. Reddy. Providing QoS Guarantees for Disk I/O. Technical Report TAMU-ECE97-02, Department of Electrical Engineering, Texas A&M University, 1997.

27 Bruce L. Worthington , Gregory R. Ganger , Yale N. Patt, Scheduling algorithms for modern disk drives, Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems, p.241-251, May 16-20, 1994, Nashville, Tennessee, United States

↑ **CITINGS** ***4***

Youjip Won , Y. S. Ryu, Handling sporadic tasks in multimedia file system, Proceedings of the eighth ACM international conference on Multimedia, p.462-464, October 2000, Marina del Rey, California, United States

Ketil Lund , Vera Goebel, Adaptive disk scheduling in a multimedia DBMS, Proceedings of the eleventh ACM international conference on Multimedia, November 02-08, 2003, Berkeley, CA, USA

Ravi Wijayaratne , A. L. Narasimha Reddy, System support for providing integrated services from networked multimedia storage servers, Proceedings of the ninth ACM international conference on Multimedia, September 30-October 05, 2001, Ottawa, Canada

Vijay Sundaram , Prashant Shenoy, Bandwidth allocation in a self-managing multimedia file server, Proceedings of the ninth ACM international conference on Multimedia, September 30-October 05, 2001, Ottawa, Canada

↑ INDEX TERMS

Primary Classification:

D. Software

↳ **D.4** OPERATING SYSTEMS

General Terms:

Design, Management, Theory, Verification

↑ Collaborative Colleagues:

Pawan Goyal:	Abhishek Chandra Haichen Chen Haichen Cheng N. G. Duffield Alok Goyal Anshuman Goyal Albert Greenberg Xingang Guo Charles R. Kalmanek Simon S. Lam	William T. Marshall Partho Mishra Dharmendra S. Modha Doug Nortz K. K. Ramakrishnan Krithi Ramamritham Sriram Rao Mohammad S. Raunak Jasleen Sahni Chia Shen	Prashant Shenoy Prashant J. Shenoy Vijay Sundaram Renu Tewari Harrick Vin Harrick M. Vin Jacobus E. van der Merive Jacobus E. van der Merwe	
Prashant Shenoy:	Banu Özden Micah Adler Manish Bhide Michael K. Bradshaw Abhishek Chandra Pavan Deolasee Venkata Duvvuri Lixin Gao Zihui Ge Weibo Gong Pawan Goyal Roderic Grupen Yang Guo	Saif Hasan Ping Ji Amol Katkar Purushottam Kulkarni Jim Kurose Jiang Lan Huan Li Xiaotao Liu Yong Liu Anuj Maheshwari Ketan Mayer-Patel Anoop Ninan Ankur Panchbudhe	Thomas Plagemann Krithi Ramamritham Rajeev Rastogi Mohammad S. Raunak Timothy Roscoe Lawrence Rowe Jasleen Sahni Subhabrata Sen Aashish Sharma Chia Shen Avi Silberschatz Aameek Singh John R. Smith	Vijay Sundaram John Sweeney Renu Tewari Don Towsley Abhishek Trivedi Bhuvan Urgaonkar Harrick Vin Harrick M. Vin Bing Wang Xiaolan Zhang
Harrick M. Vin:	Lorenzo Alvisi Tsipora Barzilai Francine Berman	Sergey Gorinsky Alok Goyal Anshuman Goyal	Simon S. Lam R. Greg Lavender James S. Mattson	Sriram Rao Sriram S. Rao Taylor L. Riché

Haichen Chen	Pawan Goyal	Jayadev Misra	Lawrence A. Rowe
Mon-Song Chen	Xingang Guo	Jayaram Mudigonda	Jasleen Kaur Sahni
Haichen Cheng	Jamie Jason	Rajat Mukherjee	Chia Shen
Young-ri Choi	Thomas Kaeppner	David Scott Page	Prashant Shenoy
Asit Dan	Dilip D. Kandlur	Edward J. Posnak	Prashant J. Shenoy
Daniel M. Dias	Jasleen Kaur	Siddhartha Rai	Prashant Jagdish
Amit Garg	Ravi Kokku	Srinivas Ramanathan	Shenoy
D. James Gemmell	Aaron Kunze	P. Venkat Rangan	Dinkar Sitaram
			Renu Tewari

Results for "(file system<in>ab) <and>redirection message" e-mail
Your search matched 0 of 1150196 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by Relevance in Descending order.

» View Session History

» New Search

Modify Search

(file system<in>ab) <and>redirection message

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revisir

Results for **"(file system<in>ab) <and>redirection"** e-mail
Your search matched **10** of **1150196** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History
» New Search

Modify Search

(file system<in>ab) <and>redirection »

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

» **Key**

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

Select **Article Information**

☐ **1. Kosha: A Peer-to-Peer Enhancement for the Network File System**
Butt, A.R.; Johnson, T.A.; Yili Zheng; Hu, Y.C.;
Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference
06-12 Nov. 2004 Page(s):51 - 51
AbstractPlus | Full Text: PDF(312 KB) **IEEE CNF**

☐ **2. Faster Web page allocation with neural networks**
Phoha, V.V.; Sitharama Iyengar, S.; Kannan, R.;
Internet Computing, IEEE
Volume 6, Issue 6, Nov.-Dec. 2002 Page(s):18 - 26
AbstractPlus | References | Full Text: PDF(418 KB) **IEEE JNL**

☐ **3. A Thin Storage Architecture for Wireless**
Srinivasan, S.H.;
Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Worksho
International Conference on
08-12 March 2005 Page(s):163 - 167
AbstractPlus | Full Text: PDF(112 KB) **IEEE CNF**

☐ **4. Proceedings of the Seventh Workshop on Hot Topics in Operating Systems**
Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on
29-30 March 1999
AbstractPlus | Full Text: PDF(108 KB) **IEEE CNF**

☐ **5. Pursuit of a scalable high performance multi-petabyte database**
Hanushevsky, A.; Nowak, M.;
Mass Storage Systems, 1999. 16th IEEE Symposium on
15-18 March 1999 Page(s):169 - 175
AbstractPlus | Full Text: PDF(560 KB) **IEEE CNF**

☐ **6. User-level remote data access in overlay metacomputers**
Siegel, J.; Lu, P.;
Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on
23-26 Sept. 2002 Page(s):480 - 483
AbstractPlus | Full Text: PDF(235 KB) **IEEE CNF**

7. Isolated program execution: an application transparent approach for executing u

programs
Liang, Z.; Venkatakrishnan, V.N.; Sekar, R.;
Computer Security Applications Conference, 2003. Proceedings. 19th Annual
8-12 Dec. 2003 Page(s):182 - 191
AbstractPlus | Full Text: PDF(351 KB) IEEE CNF

8. **An object-oriented software architecture for 3D mixed reality applications**
Piekarski, W.; Thomas, B.H.;
Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM Interna
Symposium on
7-10 Oct. 2003 Page(s):247 - 256
AbstractPlus | Full Text: PDF(497 KB) IEEE CNF

9. **SPIRAL: a client-transparent third-party transfer scheme for network attached di**
Xiaonan Ma; Narasimha Reddy, A.L.;
Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEE
Goddard Conference on
7-10 April 2003 Page(s):254 - 268
AbstractPlus | Full Text: PDF(346 KB) IEEE CNF

10. **NAS switch: a novel CIFS server virtualization**
Katsurashima, W.; Yamakawa, S.; Torii, T.; Ishikawa, J.; Kikuchi, Y.; Yamaguti, K.; Fuji
T.;
Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEE
Goddard Conference on
7-10 April 2003 Page(s):82 - 86
AbstractPlus | Full Text: PDF(253 KB) IEEE CNF

Results for **"(redirection<in>ti) <and> (file system message<in>metadata)"** e-mail
Your search matched **0** of **1150196** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History

» New Search

Modify Search

(redirection<in>ti) <and> (file system message<in>metadata)

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key	
IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revisir

US PATENT & TRADEMARK OFFICE
PATENT APPLICATION FULL TEXT AND IMAGE DATABASE

United States Patent Application *20030135511*
Kind Code **A1**
Anderson, Owen T. ; et al. **July 17, 2003**

10/044,730

Method, apparatus, and program for separate representations of file system locations from referring file systems

Abstract

A first file system includes a data object that references a second file system. The data object can be a new or existing file type with data identifying the second file system or some of its properties. The data required to locate the second file system is stored in a file system location data structure that may be located outside the first file system. The data object may then contain a key value, such as a name or a number, identifying the second file system, that can be used to look up the file system location. A referencing server may encode the file system identification and include the encoded file system identification rather than a path. When a server receives a request with a path that is encoded, the server decodes the file system identification. Then, the server may locate the root of the file system identified by the file system identification and return the root object to the client. Location of the root can be done either by accessing the file system location data structure or by using another data structure. A root referral object is the top level object in all participating file servers. It contains a referral to a root file system identification, which is the root file system. Since all participating file systems contain the same root file system identification, all clients will view the same name space regardless of which file server is initially contacted.

Inventors: **Anderson, Owen T.**; *(Chapel Hill, NC)* ; **Everhart, Craig F.**; *(Pittsburgh, PA)* ; **Shmueli, Boaz**; *(Pittsburgh, PA)*

Correspondence Name and Address:
Jeanine S. Ray-Yarletts
IBM Corporation T81/503
PO Box 12195
Research Triangle Park
NC
27709
US

Assignee Name and Adress:
International Business Machines Corporation
Armonk

NY

Serial No.: **044730**
Series Code: **10**
Filed: **January 11, 2002**

U.S. Current Class: **707/101**
U.S. Class at Publication: **707/101**
Intern'l Class: G06F 007/00

Claims

What is claimed is:

1. A method, in a requested file system server, for servicing a request, comprising: receiving a request for a referencing object from a client, wherein the referencing object refers to a referenced file system; looking up a location of the referenced file system in a separate data structure; and returning a redirection message indicating the location of the referenced file system to the client.

2. The method of claim 1, wherein the redirection message includes an address of a referenced file system server.

3. The method of claim 2, wherein the redirection message further includes a path.

4. The method of claim 2, wherein the referencing object has a file system identifier.

5. The method of claim 4, further comprising: encoding the file system identifier, wherein the redirection message further includes the encoded file system identifier.

6. The method of claim 5, wherein the referencing object is a top level object for a uniform namespace including all file systems on participating file system servers.

7. The method of claim 2, wherein the referenced file system server is the requested file system server.

8. The method of claim 1, wherein the separate data structure comprises a file system location database.

9. The method of claim 1, further comprising: receiving a redirected request for a file system object; identifying an encoded file system identifier in the redirected request; decoding the encoded file system identifier to form a file system identifier corresponding to a requested file system; looking up a path for the requested file system in a file system identifier data structure; and retrieving the root of the requested file system using the path for the requested file system.

10. The method of claim 9, wherein the file system identifier data structure comprises a file system identifier table.

11. The method of claim 9, wherein the separate data structure and the file system identifier data structure are stored in a file system location database.

12. The method of claim 1, wherein the referencing object is a top level object for a uniform namespace including all file systems on participating file system servers.

13. A method, in a requested file system server, for servicing a request, comprising: receiving a request for a file system object, wherein the request includes an encoded file system identifier; decoding the encoded file system identifier to form a file system identifier corresponding to a requested file system; looking up a path for the requested file system in a file system identifier data structure; and retrieving the root of the requested file system using the path for the requested file system.

14. The method of claim 13, wherein the file system identifier data structure is stored in a table.

15. The method of claim 13, wherein the file system identifier data structure is stored in a file system location database.

16. An apparatus, in a requested file system server, for servicing a request, comprising: receipt means for receiving a request for a referencing object from a client, wherein the referencing object refers to a referenced file system; location means for looking up a location of the referenced file system in a separate data structure; and return means for returning a redirection message indicating the location of the referenced file system to the client.

17. The apparatus of claim 16, wherein the redirection message includes an address of a referenced file system server.

18. The apparatus of claim 17, wherein the redirection message further includes a path.

19. The apparatus of claim 17, wherein the referencing object has a file system identifier.

20. The apparatus of claim 19, further comprising: encoding means for encoding the file system identifier, wherein the redirection message further includes the encoded file system identifier.

21. The apparatus of claim 20, wherein the referencing object is a top level object for a uniform namespace including all file systems on participating file system servers.

22. The apparatus of claim 17, wherein the referenced file system server is the requested file system server.

23. The apparatus of claim 16, wherein the separate data structure comprises a file system location database.

24. The apparatus of claim 16, further comprising: means for receiving a redirected request for a file system object; means for identifying an encoded file system identifier in the redirected request; means for decoding the encoded file system identifier to form a file system identifier corresponding to a requested file system; means for looking up a path for the requested file system in a file system identifier data structure; and means for retrieving the root of the requested file system using the path for the requested file system.

25. The apparatus of claim 24, wherein the file system identifier data structure comprises a file system identifier table.

26. The apparatus of claim 24, wherein the separate data structure and the file system identifier data

structure are stored in a file system location database.

27. The apparatus of claim 16, wherein the referencing object is a top level object for a uniform namespace including all file systems on participating file system servers.

28. An apparatus, in a requested file system server, for servicing a request, comprising: receipt means for receiving a request for a file system object, wherein the request includes an encoded file system identifier; decoding means for decoding the encoded file system identifier to form a file system identifier corresponding to a requested file system; path means for looking up a path for the requested file system in a file system identifier data structure; and retrieval means for retrieving the root of the requested file system using the path for the requested file system.

29. The apparatus of claim 28, wherein the file system identifier data structure is stored in a table.

30. The apparatus of claim 28, wherein the file system identifier data structure is stored in a file system location database.

31. A computer program product, in a computer readable medium, for servicing a request, comprising: instructions for receiving a request for a referencing object from a client, wherein the referencing object refers to a referenced file system; instructions for looking up a location of the referenced file system in a separate data structure; and instructions for returning a redirection message indicating the location of the referenced file system to the client.

32. A computer program product, in a computer readable medium, for servicing a request, comprising: instructions for receiving a request for a file system object, wherein the request includes an encoded file system identifier; instructions for decoding the encoded file system identifier to form a file system identifier corresponding to a requested file system; instructions for looking up a path for the requested file system in a file system identifier data structure; and instructions for retrieving the root of the requested file system using the path for the requested file system.

Description

RELATED APPLICATIONS

[0001] This application is related to commonly assigned and co-pending U.S. patent application Ser. No. 09/969,294 (Attorney Docket No. RSW920010139US1) entitled "Apparatus and Method for Offloading Application Components to Edge Servers", filed on Sep. 28, 2001; U.S. patent application Ser. No. 09/960,451 (Attorney Docket No. RSW920010141US1) entitled "Method and Apparatus for Minimizing Inconsistency Between Data Sources in a Web Content Distribution System", filed on Sep. 21, 2001; and U.S. patent application Ser. No. 09/960,448 (Attorney Docket No. RSW920010142US1) entitled "Method and Apparatus for Caching Subscribed and Non-Subscribed Content in a Network Data Processing System", filed on Sep. 21, 2001, which are hereby incorporated by reference.

BACKGROUND OF THE INVENTION

[0002] 1. Technical Field

[0003] The present invention relates to network file systems and, in particular, to supporting a uniform name space and transparent migration and replication of individual file systems. Still more particularly,

the present invention provides a method, apparatus, and program for separate representations of file system locations from the data in referring file systems.

[0004] 2. Description of Related Art

[0005] A network file system is a mechanism, an architecture, that allows a computer to use files on a separate machine as if they were local to the first machine. Network file systems include a high-level network protocol that provides the structure and language for file requests between clients and servers. The protocol provides the commands for opening, reading, writing and closing files across the network and may also provide access to directory services. In network file systems that support referrals, a client may ask a server for information about a name that appears in a first file system, as seen from the client, but is a reference to a second file system. As such, the response from the first file system must include information about the second file system, such as its location in the network. The client will then "mount" the second file system. Mounting simply means setting up the client's operating system to do input/output (I/O) operations on a file system.

[0006] FIGS. 1A-1C depict example network file systems.

[0007] Particularly, with reference to FIG. 1A, file system 106 includes a directory "usr." The "usr" directory includes a reference to file system "foo." The reference redirects a client to file system 116.

[0008] It is assumed that objects are arranged in a treelike structure, where files are arranged in directories and directories can contain other directories. Access to objects is achieved using path names, where a component of the path name designates the next sub-directory in the tree. The path starts at the top of the tree. A common convention uses forward slashes or back slashes to separate sub directories, and a single slash or backslash at the beginning of the path refers to the top of the hierarchy. For example, the path /a/b/C refers to an object "C" that is in directory "b." Directory "b" is in directory "a," which belongs at the top level of the hierarchy.

[0009] A file system may be moved from one location to another, such as to a new server. The referenced location must then redirect the client to the new location of the file system, and the source of the reference must be changed to indicate the new location of the file system as well. With reference to FIG. 1B, file system 106 includes a directory "usr." The "usr" directory includes a reference to file system "foo." The reference redirects a client to file system 116. However, "foo" has moved to file system 126. Therefore, file system 116 must include information to redirect the client to file system 126. At the same time, the information in file system 106 that had directed clients to file system 116 must be changed to direct them to file system 126.

[0010] A file system may also be replicated for increased reliability. Multiple file systems can thus reference a mounted file system. For example, in FIG. 1C, file system 106 includes a reference to file system "foo" in file system 116, file system 136 includes a reference to file system "foo" in file system 146, and file system 156 includes a reference to file system "foo" in file system 166. The file system "foo" moved to file system 126. Therefore, file systems 116, 146, 166 must be updated to include information to redirect the client to file system 126. However, too many updates results in inefficiency and a higher likelihood for error. For example, in FIG. 1C, file system 166 is not updated and the client is not redirected to the correct file system.

[0011] Similarly in FIG. 1C, the references to file system "foo" in file systems 106, 136, and 156 need to be updated so as to direct clients straight to file system 126 rather than any of 116, 146, or 166.

[0012] FIGS. 2A and 2B illustrate examples of file systems that allow mounting on remote machines.

FIG. 2A depicts an example operation using Network File System (NFS), version 4 (NFSv4), which is an example of a network file system that allows mounting on remote machines. Client 208 requests an object "X" from file system (FS) server #1 206 (step 1). However, X is a mounted file system existing on FS server #2 216 and FS server #1 206 sends a redirection message identifying FS server #2 and the path, "/a/b/c/X," used to find X (step 2). Next, client 208 uses the information in the redirection message to access /a/b/c/X on server #2 (step 3).

[0013] NFSv4 requires that each referencing server include knowledge of the location and path for each mounted file system in the references returned to its clients. A server can send a redirection message that redirects the client to the server itself. This may be useful, for example, when a file system object is moved within a server. In addition, a chain of redirection messages may be used, for example, when an object is moved more than once. Thus, using NFSv4 or similar network file systems, particularly with multiple referencing servers, the likelihood for error exists.

[0014] As another example, FIG. 2B depicts an example operation of using the DCE's Distributed File System (DCE/DFS), which is another example of a network file system that allows referrals to remote machines. Using DCE/DFS, client 208 requests an object "X" from FS server #1 206 (step 1). However, X is a mounted file system existing on FS server #2 216 and FS server #1 206 sends an indirection response including file system identifier "Y" used to find the file system (step 2). Next, client 208 requests the location of "Y" from file system (FS) location database 220 (step 3). The FS location database returns the location of Y, "FS server #2,11 to client 208 (step 4). Thereafter, client 208 uses the location of FS server #2 to request the object from FS server #2 216 (step 5).

[0015] NFSv4 and similar network file systems require that a referring server know the correct locations where to direct clients. The obvious implementation of referrals in NFSv4 and similar network file systems is to embed the locations of the referenced file systems directly in the data stored in the referencing file system. The combination of the movability of the referenced file systems and the replicability of the referencing file systems makes this a cumbersome solution: if a referencing file system is replicated to many read-only locations and a referenced file system is subsequently moved, all the instances of the referencing file systems must be updated even though they are read-only. DCE/DFS avoids this complication by storing only an identifier for the target file system in the referencing file system, so that the client looks up the current location for the file system given the file system identifier from the referencing server. It would be much less cumbersome for the client, not to mention conformant with NFSv4 and similar network file systems, if the server could handle the changing of file system locations without explicit updating of all references.

SUMMARY OF THE INVENTION

[0016] The present invention provides a file system that allows referencing between file systems which interacts well with motion and replication of file systems. A first file system includes a data object that references a second file system. The data object acts as a place holder for the second file system, but does not contain changeable data such as the name of the server containing system. The data object can be a new or existing file type with data identifying the second file system or some of its properties. The data required to locate the second file system, such as the name of a file server, is stored in a file system location data structure that is located outside the first file system. The data object then contains a key value, such as a name or a number, identifying the second file system, that can be used to look up the file system location. This allows the data in the second file system to be replicated or moved without requiring updates to the data in any redirecting or referencing servers.

[0017] The present invention also uses a basic notification mechanism to enable clients to access a uniform name space that can include all file system objects available on participating file servers. When

a referencing server sends a redirection message to a client, the redirection typically includes a server location and a path. In the present invention, the referencing server encodes the file system identification and includes the encoded file system identification rather than the path. When a server receives a request with a path that is encoded, the server decodes the file system identification. Then, the server may locate the root of the file system identified by the file system identification and return the root object to the client. Location of the root can be done either by accessing the file system location data structure or by using another data structure.

[0018] To guarantee that clients will enter the name space at the same point, and thus view the same name space regardless of the initial participating server contacted, the present invention includes a special referral object, called the root referral object, and a special root file system. The root referral object is the top level object in all participating file servers. It contains a referral to a special designated file system identification, which is the special root file system. Whenever a client contacts a participating server and asks for the root object, the server will send a redirection message to the client containing the file system identification of the root file system. Since all participating file systems contain the same root file system identification, all clients will view the same name space regardless of which file server is initially contacted.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0020] FIGS. 1A-1C depict example network file systems;

[0021] FIGS. 2A and 2B illustrate examples of file systems that allow mounting on remote machines;

[0022] FIG. 3 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented;

[0023] FIG. 4 is a block diagram of a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

[0024] FIG. 5 is a block diagram illustrating a data processing system in which the present invention may be implemented;

[0025] FIGS. 6A-6D illustrate example file systems that allow mounting on remote machines in accordance with a preferred embodiment of the present invention; and

[0026] FIG. 7 is a flowchart illustrating the operation of a file system server in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0027] With reference again to the figures, FIG. 3 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented, Network data processing system 300 is a network of computers in which the present invention may be implemented. Network data processing system 300 contains a network 302, which is the medium used to provide communications links between various devices and computers connected together within network data

processing system 300. Network 302 may include connections, such as wire, wireless communication links, or fiber optic cables.

[0028] In the depicted example, servers 304, 314, 324 are connected to network 302. Servers 304, 314, 324 serve requests for storage units 306, 316, 326, respectively. In addition, clients 308, 310, 312 are connected to network 302. These clients 308, 310, 312 may be, for example, personal computers or network computers. In the depicted example, servers 304, 314, 316 provides data stored in storage units 306, 316, 326 to clients 308-312. Clients 308, 310, 312 are clients to server 304, for example. Network data processing system 300 may include additional servers, clients, and other devices not shown.

[0029] In the depicted example, network data processing system 300 is the Internet with network 302 representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system 300 also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). FIG. 3 is intended as an example, and not as an architectural limitation for the present invention.

[0030] Referring to FIG. 4, a block diagram of a data processing system that may be implemented as a server, such as one of servers 304, 314, 324 in FIG. 3, is depicted in accordance with a preferred embodiment of the present invention. Data processing system 400 may be a symmetric multiprocessor (SMP) system including a plurality of processors 402 and 404 connected to system bus 406. Alternatively, a single processor system may be employed. Also connected to system bus 406 is memory controller/cache 408, which provides an interface to local memory 409. I/O bus bridge 410 is connected to system bus 406 and provides an interface to I/O bus 412. Memory controller/cache 408 and I/O bus bridge 410 may be integrated as depicted.

[0031] Peripheral component interconnect (PCI) bus bridge 414 connected to I/O bus 412 provides an interface to PCI local bus 416. A number of modems may be connected to PCI local bus 416. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers 308-312 in FIG. 3 may be provided through modem 418 and network adapter 420 connected to PCI local bus 416 through add-in boards.

[0032] Additional PCI bus bridges 422 and 424 provide interfaces for additional PCI local buses 426 and 428, from which additional modems or network adapters may be supported. In this manner, data processing system 400 allows connections to multiple network computers. A memory-mapped graphics adapter 430 and hard disk 432 may also be connected to I/O bus 412 as depicted, either directly or indirectly.

[0033] Those of ordinary skill in the art will appreciate that the hardware depicted in FIG. 4 may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

[0034] The data processing system depicted in FIG. 4 may be, for example, an IBM e-Server pseries system, a product of International Business Machines Corporation in Armonk, N.Y., running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

[0035] With reference now to FIG. 5, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system 500 is an example of a

client computer. Data processing system 500 employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor 502 and main memory 504 are connected to PCI local bus 506 through PCI bridge 508. PCI bridge 508 also may include an integrated memory controller and cache memory for processor 502. Additional connections to PCI local bus 506 may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter 510, SCSI host bus adapter 512, and expansion bus interface 514 are connected to PCI local bus 506 by direct component connection. In contrast, audio adapter 516, graphics adapter 518, and audio/video adapter 519 are connected to PCI local bus 506 by add-in boards inserted into expansion slots. Expansion bus interface 514 provides a connection for a keyboard and mouse adapter 520, modem 522, and additional memory 524. Small computer system interface (SCSI) host bus adapter 512 provides a connection for hard disk drive 526, tape drive 528, and CD-ROM drive 530. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

[0036] An operating system runs on processor 502 and is used to coordinate and provide control of various components within data processing system 400 in FIG. 4. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system 500. Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive 526, and may be loaded into main memory 504 for execution by processor 502.

[0037] Those of ordinary skill in the art will appreciate that the hardware in FIG. 5 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in FIG. 5. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

[0038] As another example, data processing system 500 may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system 500 comprises some type of network communication interface. As a further example, data processing system 500 may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

[0039] The depicted example in FIG. 5 and above-described examples are not meant to imply architectural limitations. For example, data processing system 500 also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system 500 also may be a kiosk or a Web appliance.

[0040] Returning to FIG. 3, server 304 provides access to storage 306. Similarly, server 314 provides access to storage 316 and server 324 provides access to storage 325. Storage 306 may store a first file system that includes a reference to a second file system stored in storage 316. In accordance with a preferred embodiment of the present invention, an reference object is placed in the first file system to serve as a place holder for the second file system. The reference object may be a new or existing file type with data indicating the second file system or some of its properties. This allows the normal directory enumeration procedures of the first file system to include a special name in a natural fashion.

[0041] With reference to FIGS. 6A-6D, example file systems are shown that allow mounting on remote machines in accordance with a preferred embodiment of the present invention. Particularly, with respect to FIG. 6A, an example is shown in which a referencing server requests location information for a referenced file system from a file system (FS) location database. Client 608 requests object "X" from FS server #1 606 (step 1). However, X is a mounted file system existing on FS server #2 616.

[0042] A referencing object may be a special file in a file system that contains an identification of a specific file system, also known as a file system identification (FSID). The FSID may be the key used to query the FS location database. The FS location database may contain, for each file system, the location of the server on which the file system resides. It may also contain the path name of the root of the file system on each server.

[0043] In the present invention, the purpose of the referencing object is to serve as a link to the root of another file system. From the client's perspective, the root of the file system that is referenced replaces the referencing object in the directory containing the referencing object. The mounted file system is thus accessed using the name of the referencing object. The client cannot itself access the referencing object using conventional file system operations. The access is manipulated by the referencing server.

[0044] FS server #1 606 sends a request for the location of x to FS location database 620 (step 2). It is convenient for the object that is in the file system on FS server #1 not to contain, itself, the information needed to the redirect the client. For example, a referencing object "X" on FS server #1 may contain a key value, such as a name or number, identifying the referenced file system.

[0045] FS server #1 606 may then use the key value to query FS location database 620 for the location of the file system.

[0046] FS location database 620 then returns the location of FS server #2 and the path "/a/b/c/X" to server 606 (step 3). Then, FS server 606 returns the location of FS server #2 and the path "/a/b/c/X" to client 608 (step 4). Client 608 then may send a request for "/a/b/c/X" to FS server #2 616 (step 5). FS server #2 may then access the file system and return the file system root to client 608. Alternatively, the location information may be found in some other data structure, such as a table in memory, rather than a database.

[0047] The key value associated with the file system may be static. Thus, the mounted file system may be moved, put off-line, replicated, copied, or cloned, and the referencing object need not be changed. Yet, the database or table in which the attributes for the referenced file system are maintained may be updated and changed, corresponding to the updates and changes of the referenced file system itself, without invalidating the data in the referencing file system.

[0048] In accordance with a preferred embodiment of the present invention, a basic notification mechanism is used to enable clients to access a uniform name space, which can include all file system objects available on all participating file servers. Thus, if a client uses a certain path name to reach a file system object using any of the file servers, any other client can use the same path name to reach the same object.

[0049] Turning to FIG. 6B, client 608 requests object "X" from FS server #1 606 (step 1). However, X is a mounted file system existing on FS server #2 616. FS server #1 606 sends a request for the location of X to FS location database 620 (step 2). FS location database 620 then returns the location of FS server #2 to server 606 (step 3).

[0050] In accordance with a preferred embodiment of the present invention, FS server #1 does not send

a real path name to the client. Rather, FS server #1 606 encodes the FSID, in this case "X," using a predetermined, system wide encoding algorithm. The encoded FSID resembles a path name and can easily be decoded back to the FSID. For example, if the FSID is a text string, such as "user.bob," a simple encoding might be to add the string "###" at the beginning and at the end of the FSID, so the path would be "###user.bob###." As such, in the example shown in FIG. 6B, the encoded FSID would be "###X###." A requirement that disallows real path names from resembling encoded FSIDs must be enforced. A person of ordinary skill in the art will recognize that other encoding methods may also be used.

[0051] In FIG. 6B, FS server 606 returns the location of FS server #2 and the path "###X###" to client 608 (step 4). The encoded FSID is transparent to the client. The client, upon receiving the name of the new server and the encoded FSID, contacts the new server and retries to access the object by passing the new server the path name given by the old server, which is the encoded FSID. Thus, client 608 then sends a request for "###X###" to FS server #2 616 (step 5). FS server #2 recognizes the path name as an encoded FSID and requests the location of X from FS location database 620 (step 6). In return, FS location database 620 returns the path "/a/b/c/X" to FS server #2 616 (step 7). FS server #2 may then access the file system and return the file system root to client 608.

[0052] In an alternative and preferred embodiment, the location of the root of each file system can be stored in a local table that maps the FSID of each file system to a local path name. With reference to FIG. 6C, client 608 requests object "X" from FS server #1 606 (step 1). However, X is a mounted file system existing on FS server #2 616. FS server #1 606 sends a request for the location of X to FS location database 620 (step 2). FS location database 620 then returns the location of FS server #2 to server 606 (step 3).

[0053] FS server 606 returns the location of FS server #2 and the path "###X###" to client 608 (step 4). Client 608 then sends a request for "###X###" to FS server #2 (step 5). FS server #2 recognizes the path name as an encoded FSID and requests the location of X from FSID table 630 and returns the path "/a/b/c/X" to FS server #2 616. FS server #2 may then access the file system and return the file system root to client 608.

[0054] Alternatively, a file system may be mounted in a special directory using a special mount point. The mount point may be a string represnetion of the FSID. In the example of FIG. 6C, the file system identified by "X" could be mounted in the "/exports/X." Therefore, FS server #2 616 need not look the FSID up in a table or database, because the file system is mounted in a predetermined directory which serves as an appropriate table.

[0055] To guarantee that clients will enter the name space at the same point, and thus view the same name space regardless of the initial participating server contacted, the present invention includes a special referencing object, called a root referencing object, and a special root file system. The root referencing object is the top level object in all participating file servers. It contains a reference to a special designated FSID, which is the identifier for the special designated root file system.

[0056] As an example, the FSID referred to by the root referencing object may be "rootfs." Whenever a client contacts a server and requests the root object, the server will access the root referencing object. Turning now to FIG. 6D, client 608 requests the top level object from FS server #1 606 (step 1). FS server #1 606 accesses the root referencing object and sends a request to FS location database 620 for the location of the file system with the FSID of "rootfs" (step 2). The FS location database returns the location of rootfs, FS server #2, to FS server #1 606 (step 3). FS server #1 then encodes the FSID to form "###rootfs###" and sends a redirection message indicating that the file system is on FS server #2 and the path is "###rootfs###" to client 608 (step 4).

[0057] Next, client 608 sends a request for "###rootfs###" to FS server #2 616 (step 5). FS server #2 recognizes the path name as an encoded FSID and sends a request for the location of the rootfs file system to FS location database 620 (step 6). The FS location database then returns the path, "/a/b/c/," to FS server #2 616 (step 7). FS server #2 may then access the file system and return the file system root to client 608. In an alternative and preferred embodiment, the location of the root of the file system can be stored in a local table that maps the FSID of each file system to a local path name.

[0058] The example in FIG. 6D shows only two file system servers. However, fewer or more participating servers may be included. For example, the network file system may include only a single server, such as server 616. If a client requests the top level object of FS server #2 616, the server may send a redirection message redirecting the client to itself with the encoded FSID, "###rootfs###." The example of FIG. 6D may also include more participating file system servers, each of which has the same referencing object as its top level object.

[0059] With reference to FIG. 7, a flowchart is shown illustrating the operation of a file system server in accordance with a preferred embodiment of the present invention. The process begins and receives a request for an object (step 702). A determination is made as to whether the object is a reference to another file system (step 704). If the object is a reference to a separate file system, the process looks up the location of the separate file system (step 706) and encodes the FSID of the file system (step 708). Next, the process returns a redirection message indicating the location and the encoded FSID as the path name to the client (step 710) and ends.

[0060] If the object is not a referencing object in step 704, a determination is made as to whether the object is a top level object (step 712). If the object is a top level object, the process accesses the root referencing object indicating an FSID of "rootfs" (step 714). Then, the process looks up the location of the file system (step 706) and encodes the FSID of the file system (step 708). Thereafter, the process returns a redirection message indicating the location and the encoded FSID as the path name to the client (step 710) and ends.

[0061] If the object is not a top level object in step 712, a determination is made as to whether the path is an encoded FSID (step 716). If the path is an encoded FSID, the process decodes the path to form the FSID (step 718), accesses and returns the root of the file system (step 720), and ends. If the path is not an encoded FSID in step 716, the process accesses and returns the real object (step 722) and ends.

[0062] A referencing object may refer to a file system on an unmodified server that does not incorporate the present invention. This may be achieved by having the referencing object itself include the location of the unmodified server and the path name of the file system on that server. When a client accesses the referencing object, the server will reply with the information included in the FS location database for the file system on the unmodified server.

[0063] The referencing object may also include an FSID, as if it referred to a modified server. However, the file system database may contain a special tag that marks the file system stored on an unmodified server, together with the server location and path name. The server may then reply to the client with the information retrieved from the FS location database. However, the path name cannot comprise an encoded FSID, and the unmodified server may not be configured to contain referrals to other file systems.

[0064] The present invention may also support file systems that are replicated on various modified servers. If the network file system protocol supports a redirection message that includes more than one server, such as is the case with NFSv4, the present invention may be extended to include the locations of

all the servers that host a particular file system. When a client attempts to access a referencing object, the modified server may reply with a list of file system server locations. The client may then select a server to access.

[0065] Alternatively, if the network file system protocol does not support a redirection message that includes more than one server location, the modified server, after receiving the list of server locations from the FS location database, may choose which server to include in the redirection message. The selection algorithm may be, for example, a round robin algorithm, and may be based on various factors, such as server load and server response time.

[0066] Thus, the present invention solves the disadvantages of the prior art by providing a data object that references a second file system that is interpreted directly by the file server. The data required to locate the second file system is stored in a file system location data structure that may be located outside the first file system. The data object may then contain a key value, such as a name or a number, identifying the second file system, that can be used to look up the file system location. Therefore, a file system may be referred to by another file system transparently to the client.

[0067] A referencing server encodes the file system identification and includes the encoded file system identification rather than a path. When a server receives a request with a path that is encoded, the server decodes the file system identification. Then, the server may locate the root of the file system identified by the file system identification and return the root object to the client. Location of the root can be done either by accessing the file system location database structure or by using another, local data structure. A root referral object is the top level object in all participating file servers. It contains a referral to a root file system identification, which is the root file system. Since all participating file servers contain the same root file system identification, all clients will view the same name space regardless of which file server is initially contacted.

[0068] It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMS, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

[0069] The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

* * * * *